Topics:

1. Class (.h and .cpp)
2. String
3. Loop(s)
4. array
5. Static variables
6. Defining function as const
7. Operator Overloading

Filename: Stocks.h

```
#ifndef STOCKS_H
#define STOCKS_H

#include <string>
using namespace std;

class Stocks {
    public:
        Stocks();
        Stocks(string, double);

        void setStockName(string);
        void setPrice(double);
        string getStockName() const;
        double getPrice() const;

        // operator overloading
        void operator+= (Stocks) ; // we want to add all the prices

    private:
        string stockName;
        double price;
};

#endif
```

Filename: Stocks.cpp

```cpp
#include "Stocks.h"
#include <iostream>

 Stocks::Stocks() {
    setStockName("");
    setPrice(0);
 }

 Stocks::Stocks(string s , double d) {
     setStockName(s);
     setPrice(d);

 }

void Stocks::setStockName(string s) {
    stockName = s;
}

void Stocks::setPrice(double p) {
    price = p;
}

string Stocks::getStockName() const {
    return stockName;
}

double Stocks::getPrice() const {
    return price;
}

// operator overloading
void Stocks::operator+=(Stocks stk) {
    // cout is just to show who owns the stock

    cout << "============================" << endl;
    cout << stockName << " : " << price << endl;
    cout << stk.getStockName() << " : " << stk.getPrice() << endl;

    // this is the actual process
    stockName += stk.getStockName() +  ", ";
    price += stk.getPrice();

}
```

Filename: FileUtil.h

```h
#ifndef FILEUTIL_H
#define FILEUTIL_H

#include <string>
#include "Stocks.h"

using namespace std;

class FileUtility {
    public:
        static void openRead(string, Stocks [], int size, string h[], int h_size);


    private:


};

#endif
```

Filename: FileUtil.cpp

```cpp
#include <fstream>

#include "FileUtil.h"

#include <iostream>
#include <string>
using namespace std;

void FileUtility::openRead(string name, Stocks data[], int size, string header[], int h_size) {
    ifstream input;
    string line;

    // open file
    input.open(name);

    // getting header
    getline(input,line);
    for (int x=0, pos ; x < h_size; x++) {
        pos = line.find(",");
        header[x] = line.substr(0,pos > 0 ? pos : line.length());
        line.erase(0,pos+1);
    }

    // getting the data and into the class
    int index=0;
    int pos;
    string sname;
    double prc;

    while( getline(input,line) && index < size) {
        pos = line.find(",");
        sname = line.substr(0,pos);
        prc = stod(line.substr(pos+2));

        data[index] = Stocks(sname,prc);
        index++;
    }


    input.close(); // closes the file

}
```

Filename: TestMain.cpp

```cpp
#include "Stocks.h"
#include "FileUtil.h"

#include <iostream>
#include <iomanip>  // for setw
using namespace std;

void display(string h[], int hsize, Stocks[], int size);


int main() {
    const int SIZE = 10;
    const int HEADER_SIZE = 2;


    Stocks mystocks[SIZE];
    string header[HEADER_SIZE];

    // open the file
    FileUtility::openRead("lab5_data.csv", mystocks, SIZE, header, HEADER_SIZE);

    display(header, HEADER_SIZE, mystocks, SIZE);

    // adding prices
    Stocks temp;
    for (int x=0 ; x < SIZE; x++)
        temp += mystocks[x] ;

    cout << endl;
    cout << "Total price: " << temp.getPrice() << endl;

    return 0;

}


void display(string h[], int hsize, Stocks data[], int size) {

    for (int x=0 ; x < hsize; x++)
        cout << setw(15) << h[x];

    cout << endl;

    for (int x=0 ; x < size; x++) {
        cout << setw(15) << data[x].getStockName()  << setw(15) << fixed << setprecision(2) <<  data[x].getPrice()   << endl;
    }

}
```

Filename: Makefile

```
OBJS = Stocks.o FileUtil.o TestMain.o

StockTest: $(OBJS)
    g++ -o StockTest $(OBJS)
    rm -f $(OBJS)

Stocks.o:
    g++ -c Stocks.cpp

FileUtil.o:
    g++ -c FileUtil.cpp

TestMain.o:
    g++ -c TestMain.cpp

clean:
    rm -f core StockTest $(OBJS)
```

Requirement:

- Add 1 operator overloading of your choice (in class Stocks)
    - Provide a short description on what it is supposed to do
    - Modify the int main to call/use the operator you created
- Grading
    - 1 mark: Created an operator overloading and is working (inside int main)
    - .5 mark: created an operator overloading, compiles, but never tested
    - .0 mark